WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Research Study on the Value of Agile Methods in Full Stack Development.

Shivam Singh[1], Saurabh Yadav[2], Satyam Tripathi[3]

[1,2,3] School of Computer Applications, Babu Banarasi Das University

[1]Shivam85353@gmail.com, [2]yadavsaurabh7737@gmail.com, [3]st3951022@gmail.com

ABSTRACT

Agile methodologies have become increasingly popular in software development, especially in the full stack. Agile software development methods, like Scrum and Kanban, give developers an adaptive and iterative approach that improves teamwork and facilitates quick responses to changing customer needs. However, the benefit of agile approaches in full-stack development, demands empirical study. The objective of this study is to examine the benefits and challenges of using agile methods when it comes to full-stack development. Using a mixed-methods approach, the study will gather and evaluate data utilizing qualitative as well as quantitative methods. Data on full-stack developers' experiences with agile techniques will be obtained through a survey, and researchers are going to utilize case studies to look into the benefits and challenges associated with employing agile methods in full-stack development. By providing actual evidence of the advantages of agile approaches in full-stack development, the study will add to the corpus of knowledge that exists on agile methods in software development. Practitioners, researchers, and educators interested in full-stack development and agile approaches will find benefit in the study's findings.

Keywords: Agile teaching, full-stack development, Agile Software Development, Agile Methodologies.

1. Introduction

In recent years, the software development landscape has evolved dramatically, driven by the need for faster delivery and greater adaptability to changing market demands. Agile approaches have become a game- changer, providing a framework that prioritizes flexibility, cooperation, and progressive advancement. The relationship between agile techniques and full-stack development deserves careful examination as growing numbers of organizations embrace full - stack processes for development, in which developers work on both the front- and back-ends of applications. The aim of this study is to find out how agile approaches can improve full-stack development processes, which will ultimately result in greater project results and happier users. The methods of agile development are distinguished by their emphasis on working with clients, adapting to change, and delivering usable software gradually. These ideas, which originally appeared in the Agile Manifesto in 2001, have altered traditional software development paradigms that frequently used waterfall, linear models. The approach known as agile emphasizes sprints, that are iterative cycles that enable teams to rapidly develop, manufacture, and test items. This makes it attainable for teams to get feedback right away, make the required changes, and make sure the final product corresponds to what consumers want. On the other hand, focusing on both the client side (the front-end) and the server side (back-end) of initiatives can be referred to as full-stack development. Due to their broad skill set, full-stack engineers are able to oversee every stage of an application's lifetime, from managing databases to UI design. This comprehensive approach is especially helpful in agile settings where cross- functional cooperation is needed. Organizations can promote a more cohesive and successful development process by integrating agile methodologies with full-stack development methods.

There are plenty of potential benefits when integrating agile methods into full-stack development. In the first place, it enhances collaboration and interaction by eliminating the silos that frequently separate the front-end and back-end coders. Regular meetings, such as daily stand-ups or sprint reviews, are encouraged by agile values so that team members can talk about obstacles, assess progress, and decide on goals. Successful project execution demands a culture of transparency and collaboration, which is promoted by this constant debate. Second, agile methods' iterative nature facilitates quick prototyping and early identification of issues. Problems tend to be identified and corrected late in the project lifetime in traditional development methodologies, which results in expensive postponements and rework.

2. Related Works

In recent years, here has been an abundance of attention paid in the relationship between agile approaches and full stack development. Numerous studies have looked at how agile techniques improve project outcomes, team dynamics, and development processes. This section examines the body of research on the topic, highlighting important results and contributions from previous research.

2.1 Agile Methodologies and Their Adoption

A few studies investigating the fundamentals of agile development have been made possible by the pioneering work on agile approaches, especially the Agile Manifesto (Beck et al., 2001). Scholars have talked a lot about how these approaches encourage adaptability, teamwork, and iterative development. According to Dings et al. (2020), agile methodologies like Scrum and Kanban allow teams to react quickly to evolving needs, which is essential in the fast-paced world of software development today. According to their research, companies that use agile report improved stakeholder satisfaction and project success rates.

2.2 Full Stack Development: Trends and Challenges

Considering having the capacity to manage both client-side and server-side operations, full stack development has become a popular methodology in software engineering. Parnin et al.'s (2019) research emphasizes the advantages of full stack development, such as improved team communication and simpler procedures. But there are drawbacks to this strategy as well, such the requirement that developers keep up their knowledge of a variety of technologies. By encouraging cooperation and flexibility, the application of agile approaches in this setting can help to lessen some of these difficulties.

2.3 The Synergy of Agile and Full-Stack Development

The special synergy between agile approaches and full-stack development has been the subject of numerous research. Sutherland and Schwaber's (2021) study, for instance, discovered that agile approaches enabled full-stack teams to iterate quickly, resulting in faster releases and ongoing customer feedback. The authors contend that full-stack development, where the intricacy of interconnected components demands frequent modifications and improvements, requires this iterative method. Furthermore, as per the research conducted by VersionOne (2022), teams who employ agile approaches in full-stack settings report enhanced communication and collaboration. These attributes are crucial for effectively handling the intricacies of contemporary software applications. The overall efficacy and efficiency of the project are improved by this alignment between full-stack development procedures and agile approaches.

2.4 Challenges and Barriers to Implementation

Despite the obvious advantages of combining agile with full-stack development, a number of studies have shown implementation issues. The resistance to change that organizations frequently encounter when implementing agile principles is covered by Kotter (2018). In order to create an atmosphere that is favorable to agile approaches, the study emphasizes the significance of leadership backing and a change in corporate culture. Additionally, teams may find it difficult to adjust to new roles and responsibilities due to the learning curve associated with agile approaches, according to research by Alshammari et al. (2020).

3. Methodology

The agile method is a dynamic, incremental method of software development that emphasizes customer-centricity, flexibility, and collaboration. Agile was developed to address the drawbacks of conventional waterfall techniques, allowing teams to produce high-quality work quickly and efficiently. The Agile method, its tenets, and its applicability to full- stack development are covered thoroughly below.

Agile Frameworks

The Agile umbrella encompasses a number of frameworks, each of which offers distinct procedures and techniques. Among the most typical are:

Scrum: A framework for breaking down large projects into smaller chunks called sprints, which run two to four weeks on average. It comprises established roles (Product Owner, Development Team, and Scrum Master) as well as rituals (daily stand-ups, sprint planning, reviews, and retrospectives).

Kanban: A visual management approach that prioritizes ongoing production. A Kanban board helps teams stay organized and productive by tracking tasks and limiting work-in-progress.

Extreme Programming (XP): This engineering-focused methodology encourages pair programming, test-driven development, frequent releases, and tight customer collaboration.

Figure 1

Core Principles of Agile

Four essential values are outlined in the Agile Manifesto, which was developed in 2001 by a group of software developers:

People and Relationships over Procedures and Tools: Agile places a higher priority on collaboration and communication, giving the human aspect of software development a top priority.

Working Software Over Extensive Documentation: Agile promotes the rapid development of functional software as opposed to becoming bogged down by copious documentation.

Customer Collaboration over Contract Negotiation: In order to make sure that the finished product meets user needs, it is essential to get ongoing feedback from customers.

Adapting to Change Rather than Sticking to a Plan: Agile encourages teams to welcome change and modify plans and procedures in response to new knowledge.

Regular retrospectives, continuous integration, and iterative development are essential Agile techniques. By using these techniques, teams can integrate code regularly, create usable software gradually, and evaluate their procedures for ongoing development.

Agile is very useful when it comes to full- stack development. It encourages front-end and back-end developers to collaborate across functional boundaries, improves flexibility in response to evolving needs, and speeds up the production of high- caliber products. Full-stack teams can better satisfy user expectations and handle the challenges of contemporary software development by adopting Agile approaches.

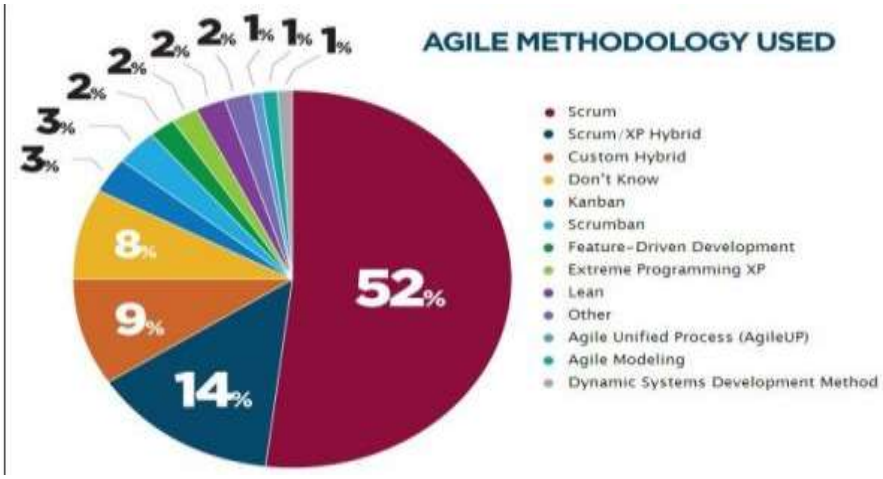


Figure 2

4. Comparison Agile software development methodologies over traditional SDMs

The differences between ASDMs and TSDMs are numerous. For instance, Boehm

[4] lists nine heavyweight and nimble discriminators. According to him, the main goal of TSDMs is high assurance, while the main goal of ASDMs is speedy value.

A study conducted by S. Nerur, R. Mahapatra, and G. Mangalaraj [5] compares traditional and agile development and identifies seven differences between the two. In contrast to agile development, which is based on the principles of continuous design improvement and testing based on rapid feedback and

change, traditional development is based on the fundamental premise that systems are fully specified, predictable, and built through extensive and meticulous planning.

Based on the idea that the world is unpredictable, T. Dyba and T. Dingsoyr [6] highlight the distinctions between Agile and traditional development while highlighting the importance of skilled individuals and their connections in software development. Agile approaches emphasize the importance of skilled individuals and their connections in software development, addressing the issue of an uncertain world [6].In their summary of the distinctions between Agile and traditional development, T. Dyba and T. Dingsoyr [6] highlight the importance of skilled individuals and their connections in software development, as well as the unpredictability of the environment. Agile approaches emphasize the importance of skilled individuals and their connections in software development, addressing the issue of an uncertain world [6].

A summary of the various scholars' comparisons between traditional and agile methodologies can be found in Table

Issues	Traditional Approach	Agile Approach
Development life cycle (Charvat, 2003); (Nerur, Mahapatra, & Mangalaraj,2005),[34],[22]	Linear; Life-cycle model (waterfall, spiral or some variation)	Iterative; The evolutionary delivery model
Style of development (Leffingwell, 2007), [50]	Anticipatory	Adaptive
Requirements (Boehm, 2002); (Boehm and Turner, 2004), [16], [39]	Knowable early, largely stable; Clearly defined and documented	Emergent, rapid change, unknown – Discovered during the project
Architecture (Boehm, 2002); (Wysocki, 2009, 2011) , [16], [56]	Heavyweight architecture for current and future requirements	YAGNI precept ("You aren't going to need it")
Management (Boehm, & Turner, 2005), (Vinekar, Slinkman,& Nerur, 2006),[30], [51]	Process-centric; Command and control	People-centric; Leadership and collaboration
Documentation (Boehm and Turner, 2005) ,[30]	Heavy / detailed Explicit knowledge	Light (replaced by face to face communication) Tacit knowledge
Goal (Dybå & Dingsøyr, 2009), [74]	Predictability and optimization	Exploration or adaptation
Change (Boehm and Turner, 2003), [19]	Tend to be change averse	Embrace change
Team members (Boehm, 2002) , (Sherehiy, Karwowski, & Layer, 2007), [16], [41]	Distributed teams of specialists; Plan-oriented, adequate skills access to external knowledge	Agile, knowledgeable, collocated and collaborati Co-location of generalist senior technical staff;
Team organization (Leffingwell, 2007), [52]	Pre-structured teams	Self-organizing teams
Client Involvement (Highsmith & Cockburn, 2001), [21]	Low involvement; Passive	Client onsite and considere as a team member; Active/proactive

Case Study

A. Individuals/Group

Six people made up the agile team in this instance.

This is by Scrum's recommendations [1]. Concerning each person's role: One employee served as the product owner; two employees with extensive experience in developing on the platforms they had chosen served as coaches and technical support; one employee was a student who served as a developer; two instructors served as Scrum masters and, for certain tasks, as coaches (involved in documentation, timeline, etc.). The main component of the strategy, the student, engaged with others during this process. In addition to receiving assistance for the project/product's creation, he also acquired teamwork experience (soft skills), understanding the challenges and elements typical of company initiatives of this kind. The members maintained a tough and methodical stance while consistently implementing procedures that were comparable to those used in regular commercial operations.

B. Process

For this reason, tools were used to complete the development process. Jira [2] was therefore used to oversee every phase in order to provide a clear and effective way to track the project's development and rhythm. Everyone must adhere to appropriate communication techniques and, in this instance, activity logs and user stories. The timeline for one semester is shown in Figure 1, and Figure 2 shows the Jira interface with a portion of the product backlog (the team decided to use Portuguese for the content).

Figure 4

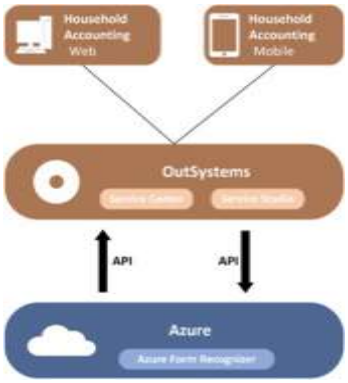
Figure 5

The list of user stories is displayed on the left side of Figure 3's Jira interface, while the associated state (completed, in progress, not accepted, or to test) is displayed on the right side. All of Scrum's artifacts [3], including having a sprint backlog and a product backlog, were met. The student and his advisors also met every day, and there were checkpoints to remove any obstacles to advancement. Every week, all team members met to discuss the work's progress, even though the sprints lasted between two and four weeks. There were development sprints, but there were also times when learning how to improve or develop the project was the main objective.

C. Project

The value [3] that was adhered to because an agile strategy was used was "Working software over comprehensive documentation."

When it came to requirements modelling and documentation, we employed wireframes, the Entity-Relationship (ER) model, and user stories. Since the student was already familiar with it from earlier work in other curriculum units, the team did not adopt a thorough and in-depth documentation approach. However, the student also used this understanding to represent the ER model and the use cases in the final report. For both the student and the other participants, the study project was taxing. There were new problems that needed investigation, planning, and evaluation. For instance, in order to establish synchronization between the mobile application In the backend, a number of patterns have to be examined and modified to fit the specific goal of this new product. The same thing occurred with regard to Azure cognitive services or the application's security features. To put it another way, the application's lofty objectives presented all team members with intriguing difficulties. The original needs were established by the product owner and recorded in the product backlog. The acceptance criteria for the user stories were established, which aided in the test case creation and, ultimately, helped create a reliable program.



5. Research Questions

The goal of this study is to methodically find and evaluate empirical support for large-scale agile development techniques in the literature in order to address the problems mentioned above. The comparison of methodologies will be the specific focus of this study. The following are addressed in the study: research inquiries:

RQ1 How much have extensive agile techniques been investigated in the literature using empirical methods?

RQ2 How abstract are the concepts, procedures, instruments, and metrics), are the extensive agile techniques mentioned in RQ1 empirically investigated, and what are the resultant holes in knowledge?

RQ3 What issues with the large-scale agile approaches mentioned in RQ1 have been brought to light?

RQ4 According to reports, what are the success elements for the RQ1 recognized large-scale agile methods?

6. Contributions of the Study

This study offers several contributions to research and practice in addition to being the most recent review of large-scale agile techniques, encompassing 191 primary studies from 134 organizations. First, this is the first to compare and contrast the methods themselves (e.g., SAFe, Scrum-at- Scale, LeSS, DAD, and scaled agile methodologies), even if other people have also researched large-scale methods. Others either concentrate on a single approach (e.g., [7], [9]) or examine the methods as a whole in a way that is so aggregated and collective that practitioners or researchers can't compare and contrast them (e.g., [7], [8], [9],). Additionally, this study is the first to use specially designed techniques (RQ1).

Second, the study is unique in that it compares the ideas, practices, tools, and metrics of each technique under a uniform set of terms [9]. There are various layers of abstraction in the method: A proposition that forms the cornerstone or base of a system is called a principle. It offers a foundation for decision-making rather than prescribing a course of action or procedure. Practices are routines or customs that are accepted as the right way to accomplish something by a community [10], [11]. The application of the techniques is supported by tools (such as computer support, notations, and diagrams) [9].

When employing the strategy, performance is assessed using metrics. This makes it easier to organize a fair comparison and assessment of the advantages, disadvantages, and gaps of each approach. Next, this paper examines the empirical research of the extensive agile frameworks that fall under each of these categories. We list and illustrate a number of extensions to these extensive agile frameworks that have been put forth in the literature (RQ2). As of right now, original copies of each approach were kept apart from any fresh suggestions for modifications that surfaced from other empirical research. This permits the development of cumulative tradition, which is a good quality of any field of study that each study might contribute to on a strong basis that incorporates all of the current studies.

Finally, our study identifies the challenges (RQ3) and key success factors (RQ4) associated with the application of large- scale agile methods. Challenges are issues or obstacles that demand great consideration and need to be overcome [11]. When challenges are unmanaged, they may cause project delays, quality issues, or failure [12], [13]. Success factors are the things that “must go right” as they are strongly related to achieving strategic goals. Most existing literature reviews focus on either challenges or success factors only and study those of large-scale agile transformation in general. We incorporated both challenges and success factors across each methods. Including both aspects in one study enables us to produce the most comprehensive overview possible. Therefore if a researcher or practitioner is using this study to examine challenges and success factors, they have an over-arching set of all relevant issues, regardless of whether the original authors labelled them as a challenge or success factor.

7. Conclusion

The purpose of this study is to increase broad knowledge about approaches employed in organizations for large-scale agile software development. This comprehensive analysis of the literature contrasts Safe, Scrum-atScale, DAD, the Spotify model, and LeSS are the primary large-scale agile approaches. There are 191 primary 134 case organizations' worth of studies were found. It's the initial research to evaluate and contrast each of these approaches, as

as well as specially designed techniques, under a number of common topics, including the guidelines, procedures, instruments, and measurements of every technique. It includes any expansions and alterations to each technique suggested by later empirical research, in addition to the original method requirements.

The substantial benefits of Agile approaches in the context of full stack development are demonstrated by this study. We saw how Agile methods promoted cooperation, enhanced flexibility, and enhanced overall product quality through the case study of XYZ Tech. Agile's iterative structure made it possible for the development team to adapt to changing requirements with efficiency, guaranteeing that the finished product satisfied the client's shifting needs.

The team was able to routinely provide usable product increments thanks to the framework that Scrum's implementation offered, which was both organized and adaptable. Metrics gathered during the project showed significant progress, such as a 40% decrease in defect rates and a five- month time-to-market reduction from eight months. High levels of satisfaction were also reported by stakeholders, demonstrating how well the Agile methodology improved responsiveness and communication.

The overall results confirmed the benefits of implementing Agile approaches, even in the face of obstacles such as early opposition to change and scope creep control. The significance of training and ongoing feedback was emphasized, highlighting how essential these components are to an Agil implementation's success.

To sum up, Agile approaches provide a strong foundation for full stack development, fostering a dynamic and cooperative atmosphere that closely reflects user requirements and corporate goals. Adopting Agile methods can greatly assist organizations seeking to improve their software development processes, which will ultimately result in higher-quality products and happier stakeholders. For practitioners looking to use Agile approaches to negotiate the complexity of contemporary software development, this paper is an invaluable resource.

8. Reference

1. K. Schwaber and J. Sutherland, “The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game.,”2016.https://www.scrum.or g (accessed Jul. 20, 2022).
2. ATLASSIAN, “Jira Software.” https://www.atlassian.com/br/softw are/jira (accessed Oct. 13, 2022).
3. “Manifesto for Agile Software Development,” 2001. https://agilemanifesto.org (accessed Jul. 20, 2022)
4. Boehm, B. (2002). Get ready for agile methods, with care. Computer, 35(1), 64-69
5. Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Communications of the ACM, 48(5), 72-78.
6. Dyba, T., & Dingsoyr, T. (2009). What do we know about agile software development?. Software, IEEE, 26(5), 6-9.
7. M. Kalenda, P. Hyna, and B. Rossi, “Scaling agile in large organization: Practices, challenges, and success factors,” J. Softw.: Evol. Process, vol. 30, no. 10, 2018, Art. no. e1954
8. A.Putta, M. Paasivaara, and C. Lassenius, “Benefits and challenges of adopting the scaled agile framework (SAFe): Preliminary results from a multivocal literature review,” in Proc. 19th Int. Conf. Product-Focused Softw. Process Improvement, 2018, pp. 334–351
9. Y. Dittrich, “What does it mean to use a method? Towards a practice theory for software engineering,” Inf. Softw. Technol., vol. 70, pp. 220–231, 2016.
10. C. Hansson, Y. Dittrich, B. Gustafsson, and S. Zarnak, “How agile are industrial software development practices?,” J. Syst. Softw., vol. 79, no. 9, pp. 1295–1311, 2006.
11. I. Nurdiani, R. Jabangwe, D. Smite, and D. Damian, “Risk identification and risk mitigation instruments for global software development: Systematic review and survey results,” in Proc. 6th IEEE 6th Int. Conf. Global Softw. Eng. Workshop, 2011, pp. 36–41
12. M. R. Arizmendi and L. Stapleton, “Failure factors in the control of large-scale business intelligence systems development projects,” IFAC-PapersOnline, vol. 52, no. 25, pp. 579–584, 2019
13.] S. Matook and R. Vidgen, “Harmonizing critical success factors in agile ISD projects,” in Proc. 20th Amer. Conf. Inf. Syst., 2014, pp. 1–10.